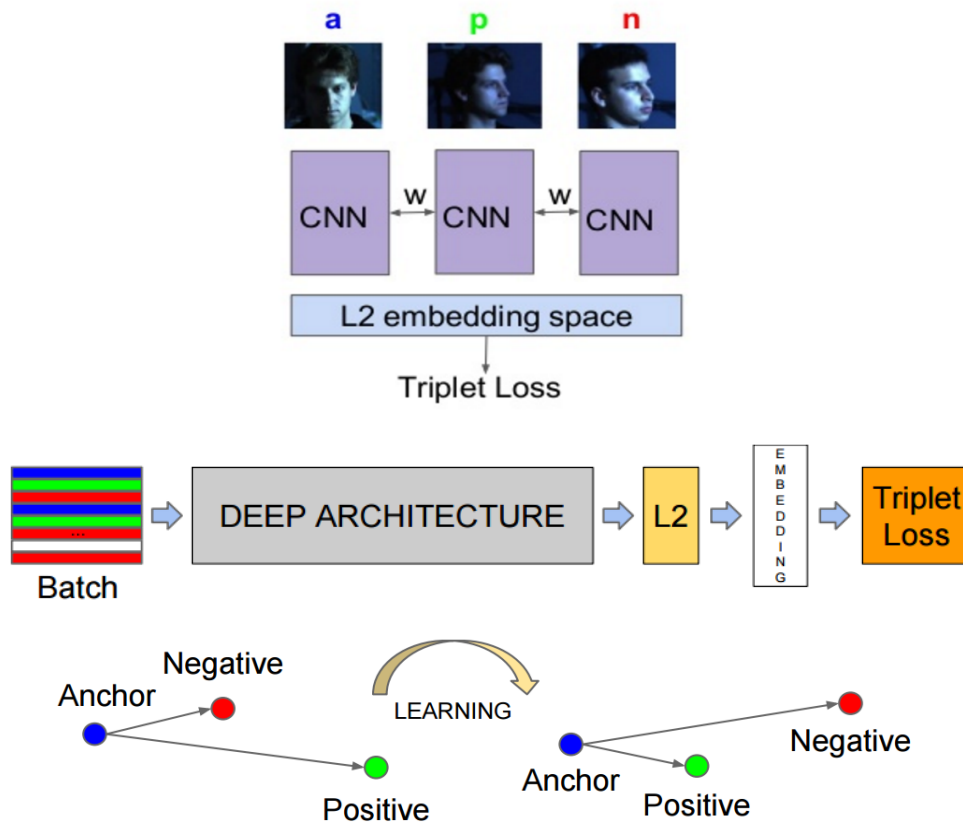# Triplet Loss

Introduction

The triplet loss is often used on person re-indentification or face recognition problems, the idea of triplet loss is to receive 3 scores, 1 anchor score, a related psoitive score and an extra negative score related to a "negative class". The training objective will minimize this loss by making related scores similar while making the difference between the anchor scores and the negative score larger.

$$Loss_{triplet} = \sum_{i=1}^{N} \left[L_{2_{distance}}\left(f_{anchor}[i] - f_{positive}[i]\right)^2 - L_{2_{distance}}\left(f_{anchor}[i] - f_{negative}[i]\right)^2 + \alpha \right]$$

The triplet loss has 3 inputs, which are the output of a neural network, a class called "anchor class", another class that is positive, and a negative class. On a face recognition problem the anchor class could be a person, and positive class could be the same person under other ilumination, and a negative class a different person.


Sometimes the inputs of the triplet loss comes from 3 different networks that share the same weights, but this also can be achieved by using a single network and building the batch with the anchor, positive and negative class. This also implies that if the triplet loss has 3 inputs during backpropagation we should return 3 gradients.

Hinge effect

Notice at the end of the Triplet loss function the "+" sign, this indicate that the loss that the function will not have negative values. This is done to avoid having a negative loss, and giving a loss of zero on the case that the distance (Anchor...negative + alpha) is bigger than (Anchor...Positive). This also will have effects on it's gradients. To illustrate better this process I will define some simple functions with this behaviour.

$$someFunc := x \rightarrow x^3 = x \rightarrow x^3 \rightarrow$$
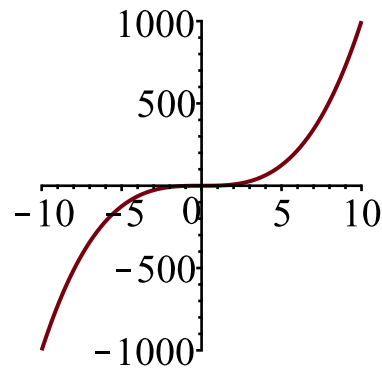
Now if we want that this function output only positive values we could say

$$someFunc_{max} := (x) \rightarrow \max(0, someFunc(x)) = x \rightarrow \max(0, someFunc(x)) \rightarrow$$
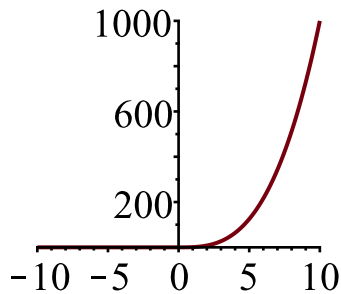
This also implies that the gradient of this functions would be different

$$\frac{d}{dx} someFunc(x) = 3\,x^2$$

$$\frac{d}{dx} someFunc_{max}(x) = \begin{cases} 0 & x \leq 0 \\ 3\,x^2 & 0 < x \end{cases}$$

Euclidian distance (Or L2 distance)

Like other distances (ie L1 distance) the L2 distance return a scalar that represent how 2 vectors(p,q) are similar to each other.

$p := [1, 2, 3] = [1, 2, 3]$

$q := [1.1, 2, 3.3] = [1.1, 2, 3.3]$

$$L_2 := (a, b, N) \rightarrow \sqrt{\sum_{i=1}^{N} (a[i] - b[i])^2} = (a, b, N) \rightarrow \sqrt{\sum_{i=1}^{N} (a_i - b_i)^2}$$

$L_2(p, q, 3) = 0.3162277660$

$$L_2(p_v, q_v, 3) = \sqrt{\left(p_{v_1} - q_{v_1}\right)^2 + \left(p_{v_2} - q_{v_2}\right)^2 + \left(p_{v_3} - q_{v_3}\right)^2}$$

On the context of loss functions we use the square l2 distance to actually cancel the squar-root term.

$$\|p_i - q_i\|_2^2 = \left(\sqrt{\sum_{i=1}^{N} (a[i] - b[i])^2}\right)^2$$
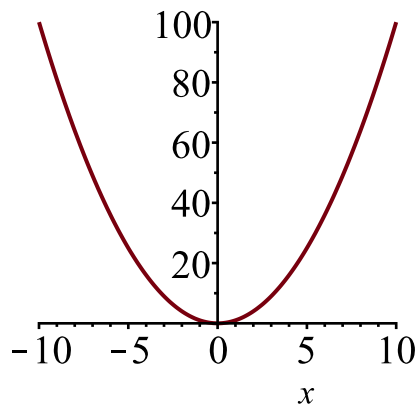
This sometimes causes confusion because now it will be called L2-norm loss

$$L_{2_{loss}} := (a, b, N) \rightarrow \sum_{i=1}^{N} (a[i] - b[i])^2 = (a, b, N) \rightarrow \sum_{i=1}^{N} (a_i - b_i)^2$$

$$L_{2_{loss}}(p, q, 3) = 0.10$$

$$L_{2_{loss}}\left(p_v, q_v, 3\right) = \left(p_{v_1} - q_{v_1}\right)^2 + \left(p_{v_2} - q_{v_2}\right)^2 + \left(p_{v_3} - q_{v_3}\right)^2$$

Also for ilustration purposes this is how the plot of the L2-norm loss looks like $L_{2_{loss_{plot}}} := x^2 = x^2 \rightarrow$

There is a variant of the L2-norm loss called MSE (Mean squared error) loss which divide the L2-norm loss by the batch-size, the idea is to decouple the batch size from the loss value.

$$MSE_{loss} := (a, b, N) \to \frac{1}{N} \cdot \left(\sum_{i=1}^{N} (a[i] - b[i])^2 \right) = (a, b, N) \to \frac{\sum_{i=1}^{N} (a_i - b_i)^2}{N}$$

$$MSE_{loss}\left(p_v, q_v, 3 \right) = \frac{1}{3} \left(p_{v_1} - q_{v_1} \right)^2 + \frac{1}{3} \left(p_{v_2} - q_{v_2} \right)^2 + \frac{1}{3} \left(p_{v_3} - q_{v_3} \right)^2$$

Get the derivative of L2-norm loss w.r.t to input

Before we delve into the gradient of the triplet-loss let's investigate how to find the derivative of the L2-norm function, by definition the idea of the norm function is to give a positive scalar number that represent the size of a vector, so we can consider the following simplification.

$$\sum_{i=1}^{N} (a[i] - b[i])^2 \Rightarrow \sum_{i=1}^{N} u^2,$$ where u will be a function that calculate the difference $(a[i] - b[i])$ vector

Let's just lay out some derivatives rules, consider u as a function

Scalar multiple rule: $\dfrac{d}{dx} (\alpha \cdot u) = \alpha \dfrac{d}{dx} u$

Sum Rule: $\dfrac{d}{dx} \sum u = \sum \dfrac{d}{dx} u$, this is nice because I don't know yet how to calculate derivatives with sum properly on maple

$$sum \cdot \frac{\partial}{\partial a} (a - b)^2 = sum (2a - 2b)$$

$$sum \cdot \frac{\partial}{\partial b} (a - b)^2 = sum (-2a + 2b)$$

Putting all together
Now let's put together the L2-norm inside the triplet loss, and also push outside all the $\sum$ operators

$f := (anchor, positive, negative, \alpha) \rightarrow [(anchor - positive)^2 - (anchor - negative)^2 + \alpha] =$

$(anchor, positive, negative, \alpha) \rightarrow [(anchor - positive)^2 - (anchor - negative)^2 + \alpha]$

$\dfrac{d}{d\ anchor} f(anchor, positive, negative, \alpha) = [-2\ positive + 2\ negative]$

$\dfrac{d}{d\ positive} f(anchor, positive, negative, \alpha) = [-2\ anchor + 2\ positive]$

$\dfrac{d}{d\ negative} f(anchor, positive, negative, \alpha) = [2\ anchor - 2\ negative]$

Now I just isolate the factors from the expression:

$factors(-2\ positive + 2\ negative) = [2, [[negative - positive, 1]]]$

$factors(-2\ anchor + 2\ positive) = [-2, [[anchor - positive, 1]]]$

$factors(2\ anchor - 2\ negative) = [2, [[anchor - negative, 1]]]$

Gradients of Triplet loss
Due to the fact that the triplet loss wants to make the distance between $[f_{anchor}, f_{positive}]$ small while making the distance $[f_{anchor}, f_{negative}]$ big by an α factor the gradients will be zero if the expression:

$\left[L2_{distance}(f_{anchor} - f_{positive})^2 - L2_{distance}(f_{anchor} - f_{negative})^2 + \alpha \right]$

gives a negative number (<=0), because it would imply that $\left[L2_{distance}(f_{anchor} - f_{positive})^2 \right]$ is already

bigger than $\left[L2_{distance}(f_{anchor} - f_{negative})^2 + \alpha \right]$

Gradient w.r.t anchor input

$\dfrac{\partial}{\partial f_{anchor}} Loss_{triplet} = \sum_{i=1}^{N} [2(f_{negative} - f_{positive})]$

Gradient w.r.t positive input

$\dfrac{\partial}{\partial f_{positive}} Loss_{triplet} = \sum_{i=1}^{N} [-2(f_{anchor} - f_{positive})]$

Gradient w.r.t negative input

$\dfrac{\partial}{\partial f_{negative}} Loss_{triplet} = \sum_{i=1}^{N} [2(f_{anchor} - f_{negative})]$

References

- https://en.wikipedia.org/wiki/Loss_function
- https://arxiv.org/pdf/1503.03832.pdf
- http://stackoverflow.com/questions/33330779/whats-the-triplet-loss-back-propagation-gradient-formula
- http://www.cnblogs.com/wangxiaocvpr/p/5452367.html
- https://en.wikipedia.org/wiki/Euclidean_distance
- http://www.slideshare.net/xavigiro/deep-learning-for-computer-vision-image-retrieval-upc-2016
- https://www.physicsforums.com/threads/derivative-of-a-function-involving-square-root-of-sum-of-squares.643123/
- http://math.stackexchange.com/questions/84331/does-this-derivation-on-differentiating-the-euclidean-norm-make-sense
- http://math.stackexchange.com/questions/883016/gradient-of-l2-norm-squared
- https://davidrosenberg.github.io/ml2015/docs/3a.loss-functions.pdf
- https://davidrosenberg.github.io/ml2015/#home
- https://devblogs.nvidia.com/parallelforall/understanding-aesthetics-deep-learning/
- http://rishy.github.io/ml/2015/07/28/l1-vs-l2-loss/
- http://nbviewer.jupyter.org/github/rishy/rishy.github.io/blob/master/ipy_notebooks/L1%20vs.%20L2%20Loss.ipynb
- http://mccormickml.com/2014/03/04/gradient-descent-derivation/
- https://www.quora.com/What-is-the-difference-between-L1-and-L2-regularization
- http://www.chioka.in/differences-between-l1-and-l2-as-loss-function-and-regularization/
- http://www.cv-foundation.org/openaccess/content_cvpr_2016/papers/Cheng_Person_Re-Identification_by_CVPR_2016_paper.pdf
- https://github.com/davidsandberg/facenet
- https://arxiv.org/pdf/1605.07270.pdf